

Laporan Praktikum Minggu Ke-4 Kontrol Cerdas

Minggu Ke-4

Nama : Muhammad Tezar Firrizqi
NIM : 224308062
Kelas : TKA 7C
Akun Github (Tautan) : <https://github.com/muhammadtezar224308062tka>

1. Judul Laporan

Reinforcement Learning for Autonomous Control

2. Tujuan Percobaan

- Memahami konsep dasar *Reinforce Learning* (RL) dalam sistem kendali
- Mengimplementasikan agen RL menggunakan algoritma *Deep Q-Network* (DQN).
- Menggunakan *OpenAI Gym* sebagai simulasi lingkungan untuk pelatihan RL.
- Melatih dan menguji agen RL untuk mengontrol lingkungan secara otonom.
- Menggunakan GitHub untuk *version control* dan dokumentasi praktikum.

3. Landasan Teori

Reinforcement Learning (RL) adalah cara belajar di mana agen belajar membuat keputusan dengan berinteraksi dengan lingkungan sekitarnya. Agen menerima reward atau penalti sebagai bentuk umpan balik atas tindakannya, sehingga dapat terus memperbaiki strategi untuk mencapai tujuan yang diinginkan. Dalam konteks sistem kendali, RL digunakan untuk memperbaiki proses pengendalian sistem tanpa harus memiliki model yang jelas, sehingga agen bisa belajar dari pengalaman langsung melalui interaksi dengan lingkungan (Norman, n.d.). Meskipun begitu, RL juga menghadapi beberapa kesulitan, seperti butuh waktu pelatihan yang cukup lama, eksplorasi yang bisa menghasilkan hasil negatif di awal, serta kesulitan dalam merancang fungsi reward yang tepat (Andreas & Kurniawan, 2018). Deep Q-Network (DQN) adalah algoritma dalam pembelajaran penguatan (*Reinforcement Learning*)

yang menggabungkan *Q-Learning* dengan jaringan saraf dalam (*deep neural networks*) digunakan untuk menyelesaikan masalah di lingkungan dengan ruang keadaan yang besar dan rumit (Putra et al., 2024).

Dengan DQN, agen mampu belajar dalam lingkungan yang kompleks, terutama ketika jumlah kemungkinan keadaan (state) sangat besar, sesuatu yang sulit dilakukan hanya dengan tabel Q seperti pada *Q-Learning* biasa. Tujuan utama agen adalah menemukan urutan tindakan yang menghasilkan hadiah terbesar dalam jangka panjang. Untuk mencapai hal ini, DQN menggunakan fungsi nilai Q yang memperkirakan seberapa baik suatu tindakan dalam kondisi tertentu. Selain *CartPole*, DQN juga bisa digunakan di lingkungan lain, seperti *LunarLander-v2*, di mana agen belajar mengendalikan pendaratan pesawat luar angkasa, atau *MountainCar-v0*, di mana agen harus mencapai puncak bukit dengan memanfaatkan momentum (Gou & Liu, 2019).

LunarLander-v2 adalah lingkungan di *OpenAI Gym* untuk menguji algoritma pembelajaran mesin, di mana agen mengendalikan pendaratan wahana luar angkasa di permukaan bulan.

Hadiah diberikan sesuai kualitas pendaratan dan penggunaan bahan bakar yang efisien, sementara penalti diberikan jika terjadi pendaratan buruk atau keluar dari area yang ditentukan. Sementara itu, *MountainCar-v0* adalah lingkungan di mana agen mengemudi mobil dan mengayunkan mobil ke kiri atau kanan untuk mendapatkan momentum agar mencapai puncak bukit. Agen bisa bergerak ke kiri, kanan, atau tidak bergerak, dengan reward negatif setiap langkah, sehingga mendorong agen untuk menyelesaikan permainan secepat mungkin sebelum mencapai batas langkah.

4. Analisis dan Diskusi

Pada mata kuliah Praktikum Kontrol Cerdas di minggu keempat ini kita melakukan percobaan yaitu implementasi *Deep Q-Network* (DQN) untuk menyelesaikan masalah *MountainCar-v0* menggunakan metode *Reinforcement learning*. Algoritma ini melatih agen agar dapat mencapai puncak bukit dengan cara memanfaatkan pengalaman yang telah dikumpulkan selama bermain. Program dimulai dengan mendefinisikan kelas DQN Agent, yang bertugas mengatur proses pembelajaran. Agen juga memiliki memory buffer berupa

Deque yang digunakan untuk menyimpan pengalaman bermain (*experience replay*), sehingga pembelajaran tidak hanya bergantung pada pengalaman terbaru, tetapi juga dari pengalaman sebelumnya. Agen membuat keputusan berdasarkan dua hal yaitu eksplorasi dan eksploitasi, yang diatur oleh parameter *Epsilon*. Awalnya, agen cenderung memilih aksi secara acak (eksplorasi), namun seiring berjalannya waktu nilai epsilon akan berkurang secara perlahan (*epsilon_decay*), sehingga agen mulai lebih sering memilih aksi berdasarkan pengalaman yang telah dimilikinya (eksploitasi). Dalam bagian utama kode, lingkungan *MountainCar-v0* dari *Gymnasium* diinisialisasi dengan mode *human render*, sehingga visualisasi pergerakan mobil dapat ditampilkan. Kemudian, agen dilatih selama episode yang telah ditentukan. Pada setiap episode, agen mengamati keadaan lingkungan (***state***), memilih aksi (***act()***), menerima umpan balik dalam bentuk reward, dan menyimpan pengalaman tersebut ke dalam *memory buffer*. Jika agen berhasil mencapai tujuan (***terminated***), ia mendapatkan reward yang lebih besar, sedangkan jika masih dalam perjalanan, ia diberikan penalti kecil untuk mendorong eksplorasi lebih lanjut. Selain itu, terdapat integrasi dengan *Pygame* untuk mendeteksi apakah pengguna menutup jendela permainan. Jika jendela ditutup, maka program akan berhenti dan *environment* akan ditutup dengan **`env.close()`**. Secara keseluruhan, kode ini menggambarkan proses pembelajaran agen menggunakan DQN, di mana agen belajar melalui pengalaman untuk mengoptimalkan pergerakan dalam lingkungan *MountainCar-v0*.

5. Assignment

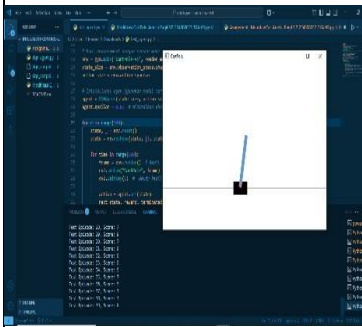
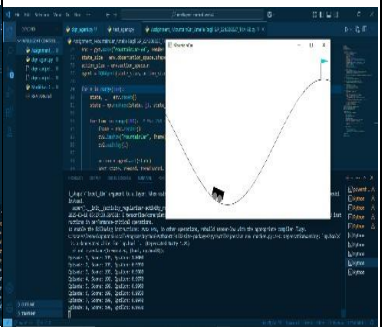
Pada hari Rabu, 24 September 2025, telah dilaksanakan praktikum tentang kontrol cerdas di Laboratorium Sistem Kendali, Laboratorium Perkeretaian, Kampus II, Politeknik Negeri Madiun. Dalam praktikum tersebut, penulis melakukan percobaan berupa Reinforcement Learning for Autonomus Control.

Dalam praktikum ini, kita membuat program yang menggunakan algoritma Deep Q-Network (DQN) untuk menyelesaikan tantangan *MountainCar-v0* dengan bantuan pustaka *Gymnasium*. Tujuannya adalah melatih agen agar mampu mencapai puncak bukit melalui pengalaman yang

diperoleh selama pelatihan. Program ini juga menerapkan konsep *experience replay*, yaitu agen menyimpan pengalaman dalam *buffer* memori dan menggunakan kembali pengalaman tersebut saat pelatihan. Hal ini membantu menjaga stabilitas pembelajaran dan mencegah agen terlalu bergantung pada pengalaman terbaru saja. Program ini memiliki beberapa fungsi yang mendukung proses pelatihan agen, seperti fungsi `act()` dalam kelas *DQNAgent*, yang digunakan untuk menentukan aksi agen berdasarkan kebijakan *epsilon-greedy*, sehingga agen tetap bisa mengeksplorasi lingkungan. Fungsi `remember()` digunakan untuk menyimpan pengalaman selama pelatihan, sedangkan fungsi `replay()` mengambil sampel pengalaman secara acak dari *buffer* memori dan memperbarui nilai Q dengan cara melatih ulang model menggunakan data tersebut. Setiap episode dimulai dengan mengatur ulang lingkungan, lalu agen memilih aksi, menerima reward, menyimpan pengalaman, dan memperbarui model sesuai dengan data yang dikumpulkan. Jika jumlah pengalaman dalam *buffer* memori sudah cukup, agen akan menjalani proses pelatihan untuk memperbaiki keputusannya. Selain itu, program ini juga menggunakan *Pygame* untuk menangani input dari pengguna. Jika jendela permainan ditutup, program akan berhenti. Namun, karena program menggunakan mode visualisasi (*human render*), visualisasi ini bisa memperlambat proses pelatihan. Oleh karena itu, bisa dipertimbangkan untuk menonaktifkan tampilan saat pelatihan dan hanya mengaktifkannya saat evaluasi. Beberapa parameter yang bisa diubah, seperti *gamma*, *epsilon decay*, atau arsitektur jaringan saraf, dapat digunakan untuk melihat dampaknya terhadap performa agen. Selain itu, metode seperti *prioritized experience replay* dapat ditambahkan untuk menjadikan proses pembelajaran lebih efisien.

6. Data dan Output Hasil Pengamatan

No.	Variabel	Hasil Pengamatan	
		CartPole-v1	MountainCar-v0
1.	Waktu Pelatihan	Lebih cepat mencapai performa optimal dalam beberapa ratus Episode.	Membutuhkan lebih banyak waktu untuk mencapai keberhasilan yang stabil.
2.	Kondisi Awal untuk Scorenya	Skor bervariasi tergantung seberapa lama tiang tetap seimbang.	Skor awal mencapai 199 karena maksimal langkah dalam satu episode adalah 200.
3.	Kondisi Akhir	Episode berakhir jika tiang jatuh atau batas langkah maksimum Tercapai.	Episode berakhir jika mobil mencapai puncak bukit (bendera).
4.	Prinsip Kerja	Sistem ini mengikuti hukum fisika, di mana jika tongkat mulai miring, gerobak harus segera bergerak ke arah kemiringan untuk menyeimbangkannya. Jadi, tongkat harus tetap seimbang di atas gerobak yang bergerak ke kiri atau kanan.	Sistem ini menampilkan sebuah mobil kecil berada di lembah di antara dua bukit dan harus mencapai puncak salah satu bukit. Mobil tidak memiliki tenaga yang cukup untuk langsung naik, sehingga harus mengandalkan momentum dengan cara mengayun bolak-balik agar dapat mencapai ketinggian yang cukup.

5.	Hasil	 <pre> function f = f(x) f = 100*(x(2)-x(1)^2)^2 + x(1)^2; end [x, fval] = fminsearch(f, [1, 1]); disp('Minimum value of f is:'); disp(fval); disp('Parameters at minimum:'); disp(x); </pre> <p>Minimum value of f is: 0.000000000000000 Parameters at minimum: 0.000000000000000 1.000000000000000</p>	 <pre> function f = f(x) f = 100*(x(2)-x(1)^2)^2 + x(1)^2; end [x, fval] = fminsearch(f, [1, 1]); disp('Minimum value of f is:'); disp(fval); disp('Parameters at minimum:'); disp(x); </pre> <p>Minimum value of f is: 0.000000000000000 Parameters at minimum: 0.000000000000000 1.000000000000000</p>
----	-------	--	---

7. Kesimpulan

Berdasarkan praktikum dan analisis yang telah dilakukan dapat disimpulkan bahwa:

- Metode *prioritized experience replay* dapat digunakan untuk membuat agen lebih cepat belajar dari pengalaman yang lebih berpengaruh terhadap keberhasilannya.
- Mode tampilan (*human render*) dalam simulasi memperlambat proses pelatihan. Oleh karena itu, saat pelatihan, sebaiknya tampilan visual dinonaktifkan dan hanya digunakan untuk evaluasi hasil.
- Lingkungan dengan *reward* yang jarang seperti *MountainCar*, membutuhkan eksplorasi yang lebih lama dan *replay buffer* yang lebih besar agar agen dapat mengumpulkan pengalaman yang cukup untuk menemukan strategi yang efektif.
- Penggunaan *target network* membantu stabilitas pembelajaran, menghindari perubahan nilai Q yang terlalu drastis.
- Dengan DQN, agen mampu belajar bahwa hanya maju terus tidak cukup untuk mencapai tujuan. Agen memahami bahwa terkadang perlu bergerak mundur terlebih dahulu untuk mengumpulkan momentum sebelum mencapai puncak.

8. Saran

Untuk meningkatkan kinerja agen dalam belajar menyelesaikan tantangan *MountainCar-v0*, ada beberapa hal yang bisa diperbaiki. Salah satunya adalah menyesuaikan parameter seperti γ , laju penurunan ϵ , dan ukuran buffer memori agar agen bisa belajar lebih efisien dan menemukan cara terbaik lebih cepat. Dari segi kecepatan, karena *MountainCar-v0* membutuhkan banyak kali pelatihan untuk memahami cara kerjanya, sebaiknya tampilan visual (*human render*) dimatikan selama proses pelatihan agar tidak menyebabkan penghambatan. Tampilan bisa diaktifkan lagi saat evaluasi untuk melihat hasil akhir dari proses belajar agen. Dengan perbaikan-perbaikan ini, diharapkan agen dapat belajar lebih cepat, lebih stabil, dan lebih efisien dalam mengatasi tantangan *MountainCar-v0*.

9. Daftar Pustaka

- Andreanus, J., & Kurniawan, A. (2018). Sejarah, Teori Dasar dan Penerapan Reinforcement Learning: Sebuah Tinjauan Pustaka. *Jurnal Telematika*, 12(2), 113–118.
<https://doi.org/10.61769/telematika.v12i2.193>
- Gou, S. Z., & Liu, Y. (2019). *DQN with model-based exploration: Efficient learning on environments with sparse rewards* (No. arXiv:1903.09295).arXiv.
<https://doi.org/10.48550/arXiv.1903.09295>
- Norman, P. (n.d.). *Training reinforcement learning model with custom OpenAI gym for IIoT scenario*.
- Putra, R. A., Syahbana, Y. A., & Ananda. (2024). Implementasi Algoritma Deep Q- Network (DQN) pada Lampu Lalu Lintas Adaptif Berdasarkan Waktu Tunggu dan Arus Kendaraan. *The Indonesian Journal of Computer Science*, 13(5).
<https://doi.org/10.33022/ijcs.v13i5.4372>